```
module alu_top (
    input wire clk,
    input wire rst,
    input wire wr_en,
    input wire [7:0] instr_in,  // 8-bit instruction input to FIFO
    input wire exec_en,           // One-cycle pulse to execute instruction from FIFO
    output [7:0] result
);
    wire [7:0] instr_out;
    wire fifo_empty, fifo_full;

    // Internal control
    wire [2:0] opcode;
    wire [4:0] imm;
    reg [7:0] acc;

    // Instantiate FIFO (Your existing FIFO)
    fifo_sync #(
        .DATA_WIDTH(8),
        .FIFO_DEPTH(16)
    ) fifo_inst (
        .clk(clk),
        .rst(rst),
        .wr_en(wr_en),
        .rd_en(exec_en),
        .din(instr_in),
        .dout(instr_out),
        .full(fifo_full),
        .empty(fifo_empty)
    );

    // Decode instruction
    assign opcode = instr_out[7:5];
    assign imm = instr_out[4:0];

    // ALU logic
    alu_8bit alu_unit (
        .opcode(opcode),
        .acc(acc),
        .imm(imm),
        .result(result)
    );

    // Accumulator update
    always @(posedge clk) begin
        if (rst)
```

```
            acc <= 0;
        else if (exec_en && !fifo_empty)
            acc <= result;
    end

endmodule
```